基于统计学特征的 android 恶意应用检测方法

冷　波[a]，李建彬[b†]

(中南大学 a. 信息科学与工程学院; b. 信息安全与大数据研究院, 长沙 410083)

摘　要：针对 Android 恶意应用检测中忽略特征统计学意义的问题，提出一种基于统计学特征的 Android 恶意应用检测方法。该方法提取应用统计学特征作为训练数据集，并采用聚类算法预处理恶意数据集以降低个体差异性对实验结果的影响。另一方面，该方法结合特征和多种机器学习算法（如线性回归、神经网络等）建立了检测模型。该方法提出的两个模型准确率均能达到 95%以上，检测时间相比于对比实验也能大幅度降低。实验结果表明，应用的统计学特征能够很好地区分良性和恶意应用，并且通过聚类算法预处理数据能够提高检测精度。

关键词：统计学特征；机器学习；个体差异性；恶意应用检测

中图分类号：TP309　　doi: 10.3969/j.issn.1001-3695.2018.03.0173

Android malicious application detection method based on statistical features

Leng Bo[a], Li Jianbin[b†]

(*a. School of Information Science & Engineering*, *b. Information Security & Big Data Research Institute*, *Central South University*, *Changsha* 410083, *China*)

Abstract: Aiming at the problem of ignoring the statistical significance of features in detection of Android malicious applications, an Android malicious application detection method based on statistical features was proposed. This method extracted the statistical characteristics of the training data set and used a clustering algorithm to preprocess the malicious data set for reducing the impact of individual differences on the experimental results. On the other hand, this method combined the features and various machine learning algorithms (such as linear regression, neural network, etc.) to establish a detection model. The accuracy rate of the two models proposed by this method could reach more than 95%, and the detection time could be greatly reduced compared with the comparison experiment. Experimental results show that the statistical characteristics of the application can be used to distinguish between benign and malicious applications, and preprocessing the data by clustering algorithm can improve the detection accuracy.

Key words: statistical features; machine learning; individual difference; malware detection.

0　引言

据 International Data Corporation (IDC)[1]报道，从 2016 年第一季度到 2017 年第一季度 Android 智能手机在全球智能手机市场份额中一枝独秀，以压倒性的优势遥遥领先其他的智能手机平台。智能手机的迅猛发展导致了人们生活发生了翻天覆地的变化。一方面智能手机已经不再是简单的通信工具，用户使用智能手机可以网上冲浪、浏览视频、购物聊天、office 办公等；另一方面智能手机也成为了用户信息的小小缩影，它包含了用户的几乎所有信息，如常见的用户身份信息、银行卡信息、通讯录等。一旦丢失或者被不法分子盗取，其后果不堪设想。同时，随着 Android 设备的日益普及，便利的网络和高价值的个人信息已经吸引了恶意软件开发者的极大兴趣。360 安全中心[2]称 Android 平台平均每天新增恶意应用 2.1 万个。因此，如何检测出 Android 恶意软件并阻止其对用户造成危害是非常迫切的。同时，这也是非常具有挑战性。目前，关于 Android 应用程序检测的研究主要集中在结合特征和机器学习算法来校验应用程序的恶意与否。通过分析 Android 应用程序的源代码或者二进制信息，恶意应用研究者提出了利用权限特征[3]、组件特征[4]、dex 特征[5]以及 smali 特征[6]进行恶意应用检测模型的训练。其中，权限特征和组件特征均来自 Android 应用的资源清单文件 AndroidManifest.xml。应用程序申请的权限和组件由特殊的标签所包含。在应用程序安装时，Android 系统会读取该文件并进行配置。Dex 特征是指从解压后应用程序的 dex 文件中解析出来的特征。Dex 文件是 Android 虚拟机 Dalvik 的执行文件并且有自己固定的格式。而 smali 特征来源于反编译应用

收稿日期：2018-03-08；**修回日期：**2018-04-25

作者简介：冷波（1992-），江西九江人，硕士研究生，主要研究方向为信息安全；李建彬（1968-），男（通信作者），教授，硕士，主要研究方向为信息安全（lijianbin@csu.edu.cn）.

程序后的 smali 文件。通过提取应用程序的静态特征构建的检测方法能够准确地进行恶意应用检测。

但是上述方法着重于理解每个特征的含义，并且忽视了个体的特殊性。同时，恶意应用的检测时间尚未受到足够的重视。因此，本文提出基于统计学特征的 Android 恶意应用检测方法，它利用权限、组件、dex 信息和 smali 信息并结合多种机器学习算法（如 SVR、MLP 等）用于恶意软件检测。该方法包括两个模型。每个模型分析和提取统计特征，充分降低个体差异性对应用检测的影响。模型 1 旨在更快地检测，而模型 2 提供更高的检测精度。本文的贡献如下：a)从崭新的角度分析 Android 应用，提取 apk 代码的统计学特征，而不是关注特征本身的意义；b)首次通过聚类算法降低个体差异性的影响，并且根据每簇大小的波动性选择聚类数目来最小化随机聚类的影响;c)使用双层学习模型以获得更好的准确性;d)根据不同场景提供两种模型。模型 1 用于更快的检测，而模型 2 用于更精确的检测。

1　相关工作

Android 恶意应用检测方法主要分为静态检测、动态检测以及动静结合的检测方法。

静态检测方法主要通过分析 Android 应用的源代码或二进制信息、提取权限、组件、函数调用、Intent、ICC（组件间通信）、操作码等内容并结合多种机器学习算法进行分析检测。权限、组件分析是指利用 AndroidManifest.xml 文件中声明的 permission、activity、service、provider、receiver 等内容进行分析[7]。其中权限代表应用程序的执行能力，能够很好地反映应用的潜在行为。而其他的组件信息是应用程序的入口，同样也能够反映应用程序的执行流程。函数调用分析是通过对应用程序源代码中使用的 API 进行结构化的处理并形成函数调用图。函数调用图从每个应用程序 MainThread 开始执行，中间能够进行复杂的逻辑处理，最后由系统保持应用程序状态。利用生成的函数调用图能够全面地反映应用程序的执行过程[8]。Android 应用中组件通信的载体是 Android Intent。Intent 能够跨越组件边界传递数据并且几乎所有的恶意应用都离不开 Intent。Feizollah 等人[9]证明了 Android Intents 作为检测恶意应用程序的特征的有效性。Idress 等人[10]利用权限和意图构建了识别 Android 恶意软件应用程序的框架。ICC（组件间通信）是指 Android 应用中的四大组件之间相互传递数据。通过分析 ICC 在应用中的使用可以进行 Android 恶意应用的检测[11]。Android 应用本质是二进制文件，是一串 0101 序列。通过分析其格式并提取操作指令可以有效地甄别恶意行为[12]。

动态分析是指利用沙盒技术，模拟用户的点击事件，自动地完成应用程序的安装、运行、卸载等操作，并记录信息以检测恶意应用的分析方法。应用程序在沙盒环境中的行为可以用于产生模型。不同的模型属于不同的类别。当应用被检测出不属于良性应用类别时就是恶意应用[13]。通过改变点击事件序列可以让应用程序产生不同的行为，并比较应用的行为以确定应用程序的恶意性[14]。系统数据在部分应用运行时会被应用所使用，研究者通过将系统数据打上标记并跟踪标记数据分析应用的信息流从而检测出应用的异常行为[15]，并且在应用运行过程中监测应用产生的网络流量也能较好地反映应用的行为[16]。部分恶意应用会检测自身是否处于真机状态从而隐藏自己的行为。因此 Salehi 等人[17]创建了一个基于主机的轻量级的检测系统，该系统在移动设备上进行检测，并可以重建应用程序的行为。

静态分析方法具有检测速率快、准确率高的特点，但不能检测出新生的恶意应用；而动态检测方法能够检测新生的恶意应用，但又特别地耗费资源。为了充分发挥两者的优势并弥补两者的不足，有研究者提出动静结合的 Android 恶意应用检测方法。动静结合的检测方法是指在恶意应用的检测过程中先使用静态分析方法进行检测，如果检测出是恶意应用，则不需要提交动态检测方法检测，否则就提交给动态分析方法进行检测[18]。

上述的几种检测方法是目前 Android 恶意应用检测研究的中心，但是这些方法都强调特征本身的内容，而忽视了特征的统计学意义，并且没有很好地解决个体差异性的影响。本文提出的方法对 Android 恶意应用检测具有一定的参考意义。

2　设计与实现

基于统计学特征的 Android 恶意应用检测方法是一种静态的检测方法。该方法包括三部分，即收集数据、提取特征和训练模型（图 1）。

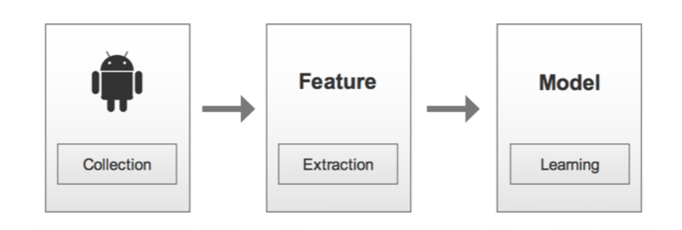


图 1　系统架构

2.1　数据收集

本文收集了大量的恶意样本和良性样本。其中恶意样本均来自 Arp D[19]数据集，该数据集包含 5 560 个恶意应用，并且属于 179 个不同的恶意软件家族，如木马、广告软件、间谍软件和信息窃取应用等。良性样本均来自 Google Play 商店。Google Play 商店是 Android 智能手机的官方应用市场，并且具有严格的应用检测制度。在 Google Play 市场上出现的应用程序必须经过开发者、应用程序代码、元数据等多重检测才能发布。所以可以认为 Google Play 上发布的应用是良性的。良性样本数据集包含来自 Google Play 中 27 个目录共 3 000 个良性应用。

2.2　提取特征

Apktool[20]可用于对 Android apk 文件进行反编译，并可轻松将资源解码为最接近的原始形式。 Unzip[21]可以用来解压

apk 文件。在本文中利用这两个工具来提取特征。通过反编译 apk 获得了大量文件夹和文件，这些文件如图 2 所示。

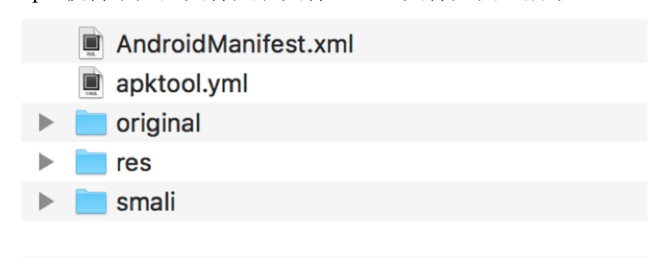


图 2　反编译示例

AndroidManifest.xml 声明应用程序申请的所有权限和组件。 Apktool.xml 包含一些附加信息，如版本和版本信息。 Res 文件夹包含不同的资源。 Smali 文件夹包含来自 Java 字节码的 smali 文件，而 original 文件夹包含一些二进制信息。图 3 显示了解压缩的 apk 文件示例。

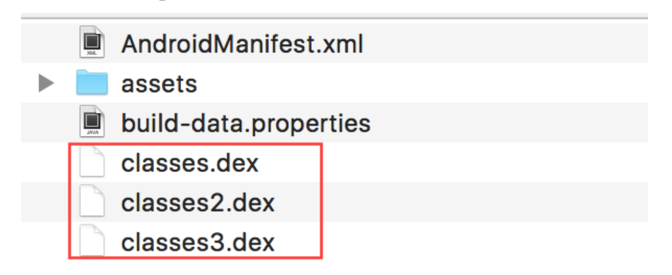


图 3　解压缩示例

本文分析了图 2 中的 AndroidManifest.xml 并提取了大量特征，如一个 apk 有多少 activity。表 1 显示了该应用程序的一个示例。

表 1　权限组件特征

特征	示例
Activity	Activity : 10
Receiver	Receiver : 5
Service	Service : 3
Action	Action : 68
Category	Category :16
Intent-filter	Intent-filter : 20
Meta-data	Meta-data : 10

DexFormat[22]列出了.dex 文件遵循的特定格式，本文通过分析其格式收集了表 2 中列出的特性。

表 2　Dex 特征

特征	示例
Link_szie	Link_size:0
Proto_ids_size	Proto_ids_size:6899
String_ids_size	Stirng_ids_size:28714
Field_ids_size	Field_ids_size:20594
Type_ids_size	Type_ids_size:5463
Method_ids_size	Method_id_size:35882

来自同一应用程序的所有 smali 文件都通过反编译存储在 smali 文件夹中。每个.smali 文件都遵循固定的语法格式。本文记录来自 smali 文件的特征信息的数量，如静态域和保护域。表 3 列出部分统计特征。

表 3　Smali 特征

特征	示例
Avg_parameter_num	Avg_paramter_num:7.2146
Min_invoke_num	Min_invoke_num:4.66666
Method_num	Method_num:42.428571286
Min_register_num	Min_register_num:0.76190

本文提取的统计学特征包括权限组件、smali 和 dex 特征。这些特征与应用程序运行密切相关，部分特征还能清晰地反映应用程序所能执行的行为。所以本文提取的特征具有一定的意义。

2.3　构建模型

图 4 所示的学习训练模块是本文的重要组成部分。

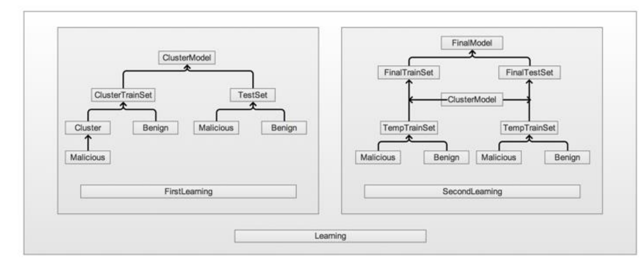


图 4　训练模块

该训练模块总共由两层组成。

第一层首先通过聚类算法降低恶意应用个体差异性对实验结果的影响。因为聚类算法本身在选取质点中心时具有随机性，所以通过多次实验的方法选取聚类数目波动性最小的聚类结果作为最终聚类结果。随后，本文在恶意应用聚类产生的每簇中加入良性样本组成训练集，并利用多种机器学习算法进行训练。所用的机器学习算法包括线性回归、支持向量机回归、神经网络、随机森林、K-近邻算法、岭回归以及贝叶斯岭回归算法。最后，本文抽取剩余数据集中的部分恶意和良性样本构建测试集来评估每个模型，并获得恶意应用每个簇每个算法的 AUC（ROC 曲线下面积）， AUC 值最大的模型就是每簇的最终模型。

在训练的第二层当中，本文首先构建了临时训练集和测试集。随后，利用第一层恶意应用每个簇模型计算临时训练集和测试集，每簇模型得到的结果并结合样本标签组成最终训练集和测试集。最后，与第一层相同，在训练集执行了多种算法，使用测试集进行测试以获得每种算法的 AUC，并决定 AUC 最佳的模型作为第二层的最终模型。

3　实验结果与分析

本文提供两种模型来满足不同场景的需求。模型 1 结合了权限、组件和 smali 特征。模型 2 结合了权限、组件和 dex 特征。以下以模型为单位分别讨论。

3.1　聚类分析

KMeans 算法是最受欢迎的聚类算法之一。其通过随机选

取质心并将样本放入一个最近质心类中来实施聚类。因此随机质心将导致聚类簇的大小不一。当质心数从 3 变化到 5 时，本文运行实验 10 次，其结果如图 5 所示。在图 5 的每个子图中，横坐标轴表示质心的数目，而纵轴表示每个簇的大小。模型 1 和 2 在四个质心时保持每个簇的大小的基本稳定。因此，聚类算法显示聚类效果最稳定的簇数大小是 4。

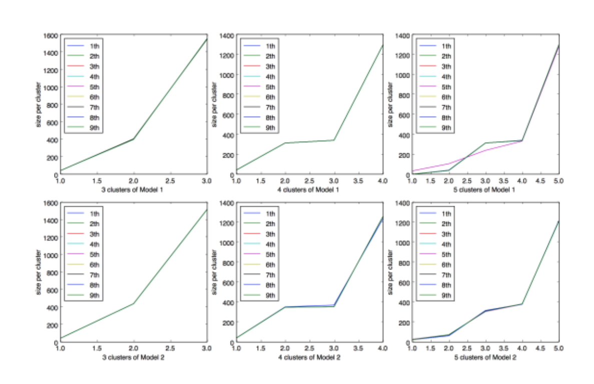


图 5　聚类分析

3.2　簇模型

基于多种算法，本文记录了每簇每个算法的真阳性率（TPR）如下：

$$TPR=\frac{TP}{TP+FN}$$
$$y_i=TPR(Cluster_i Algorithm_j)(i=1,2,3,4;j=1,2,3,4,5,6,7) \quad (1)$$

其中：TP 代表真阳性样本，阳性样本检测结果为阳性；FN 显示假阴性样本，检查为阴性但实际是阳性的样本。　另外，计算每个簇每个算法的假阳性率（FPR）如下：

$$FPR=\frac{FP}{FP+TN}$$
$$y_i=FPR(Cluster_i Algorithm_j)(i=1,2,3,4;j=1,2,3,4,5,6,7) \quad (2)$$

其中：FP 表示假阳性样本，该样本为阴性但检查为阳性；相反，TN 显示真阴性样本，阴性样本为阴性且检测结果也为阴性。

此外，本文记录了每个簇每个算法的准确度，即预测的正确结果除以所有结果。

$$Accuracy=\frac{TN+TP}{FP+TP+FN+TN}$$
$$y_i=Accuracy(Cluster_i Algorithm_j)(i=1,2,3,4;j=1,2,3,4,5,6,7) \quad (3)$$

然后，本文通过绘制 ROC 曲线获得了每簇算法的 AUC。

$$y_i=AUC(Cluster_i Algorithm_j)(i=1,2,3,4;j=1,2,3,4,5,6,7) \quad (4)$$

本文记录了最优算法的 AUC，如表 4 和 5 所示。

表 4　Dex 最优算法

Dex 簇	最优算法	AUC(曲线下面积)
第一簇	MLPRegressor	0.960402
第二簇	LinearRegression	0.967626
第三簇	LinearRegression	0.950732
第四簇	BayesianRidge	0.97165

表 5　Smali 最优算法

Smali 簇	最优算法	AUC(曲线下面积)
第一簇	MLPRegressor	0.98157
第二簇	Ridge	0.932828
第三簇	LinearRegression	0.964847
第四簇	MLPRegression	0.970324

3.3　最终模型

本文使用每个簇模型将临时训练集和测试集转换为最终集合。每个模型计算临时集以获得一列数据。由于四个簇模型，所以获得了四列，如下所示：

$$y_i=ModelCluster_i(temporary_set)(i=1,2,3,4) \quad (5)$$

然后，本文合并四列的数据和标签列如下：

$$Z=\{y_i,label\}(i=1,2,3,4) \quad (6)$$

本文对模型 1 和 2 的最终训练集和测试集执行了不同的算法。最后，实验结果记录在表 6 和 7 中。

表 6　Dex 最终算法

算法	AUC(曲线下面积)
SVR	0.9625
MLP	0.9735
RandomForest	0.9715
KNeighbors	0.9716
Ridge	0.9635
LinearRegression	0.9634
BayesianRidge	0.9635

表 7　Smali 最终算法

算法	AUC(曲线下面积)
SVR	0.9854
MLP	0.9938
RandomForest	0.9867
KNeighbors	0.9825
Ridge	0.9835
LinearRegression	0.9835
BayesianRidge	0.9835

结果显示模型 1 和 2 中 MLPRegressor 具有最好的 AUC。因此，本文使用 MLPRegressor 作为最终算法，并基于此算法构建模型。

3.4　对比分析

在 Arp D[33]的 DREBIN 中，研究人员结合 SVM（支持向量机）算法和 Android 应用程序的大量特征来区分恶意和良性应用程序。因为本文的恶意应用均来自 DREBIN 数据集当中，所以本文将该方法与本文提出的方法进行比较，并从时间效率

和准确率方面进行对比，其结果如图 6 和 7 所示。

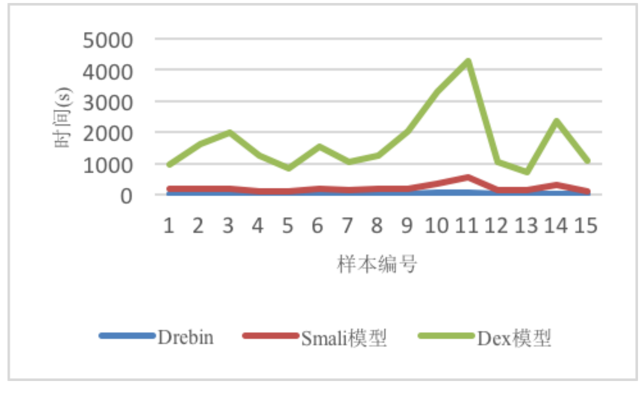


图 6　时间效率比较

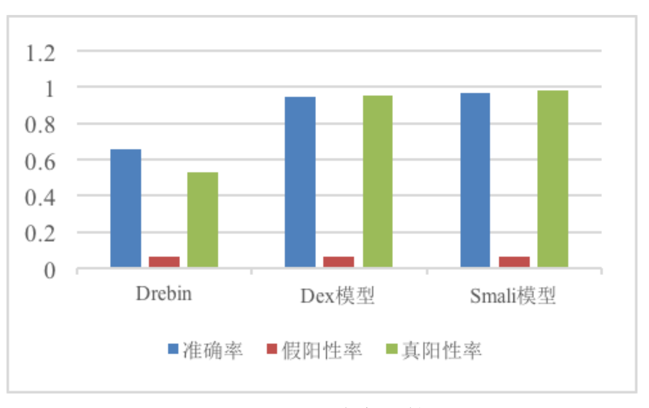


图 7　准确率比较

从图 6 可知，使用相同的测试样本，使用模型 1 显着减少了检测时间，而图 7 显示在模型 2 中获得了最高的准确度。

4　讨论

以上介绍了本文工作的创新性，实验分析以及数据结果对比分析。相比于之前的相关工作，本文能在检测效率和精度上占居一定优势，但是该方法仍然存在不足。下面简要介绍本文方法的局限性和下一步工作的方向。

本文提出的基于统计学特征的 Android 恶意应用检测方法通过提取样本中的特征进行模型训练和检测，所提取的特征维度还较低，只有四个维度。不能充分地反映统计学特征在 Android 恶意应用检测中的价值。另一方面，由于目前研究中并没有合适的良性样本集，所以本文在选取良性样本时没有做到完全的适合，而是采取其他办法解决。最后整体的样本容量也需要扩充才能进一步提高检测精度。

5　结束语

现有的工作已经提取了来自 Android 应用程序的多个方面的数据作为特征进行 Android 恶意应用检测，其中包括权限、组件、动态日志、网络流量等。但是目前的研究只是分析了特征背后所具有的实际意义，比如权限代表应用能够完成的操作，而忽视了这些特征的统计学意义，并且几乎所有的检测方法在检测时间上都有所欠缺。在本文的工作中利用聚类算法来最小化个体差异性的影响，并结合多种机器学习算法训练模型。结果显示，所提出的模型需要较少的时间来检测应用程序集，并且具有较高的检测精度。在未来的工作中，将进一步探索特征与个体之间相关性的影响，以及从不同角度提取特征，以提高 Android 恶意应用的检测准确性。

参考文献：

[1] Simon B, Melissa C, Peggy C, *et al.* Smartphone OS [EB/OL]. (2018-04-20) . https://www. idc. com/promo/smartphone-market-sh are/os.

[2] 360 烽火实验室, 360 互联网安全中心. 2017 年 Android 恶意软件年度专题报告 [EB/OL]. (2018-03-02) http://zt. 360. cn/11010 61855. php?dtd101061451&did=491056914.

[3] Liang Shuang, Du Xiaojiang. Permission-combination-based scheme for Android mobile malware detection [C]// Proc of IEEE International Conference on Communications. 2014: 2301-2306.

[4] Li Li, Bartel A, Bissyandé T F, *et al.* IccTA: detecting inter-component privacy leaks in Android apps [C]// Proc of IEEE//ACM International Conference on Software Engineering. 2015: 280-291.

[5] Maiorca D, Mercaldo F, Giacinto G, *et al.* R-PackDroid: API package-based characterization and detection of mobile ransomware [C]// Proc of Symposium on Applied Computing. 2017: 1718-1723.

[6] Tang Junjie, Cui Xingmin, Zhao Ziming, *et al.* NIVAnalyzer: a tool for automatically detecting and verifying next-intent vulnerabilities in Android apps [C]// Proc of IEEE International Conference on Software Testing, Verification and Validation. 2017: 492-499.

[7] Bhattacharya A, Goswami R T. DMDAM: data mining based detection of Android malware [C]// Proc of Icic2 Intelligent Computing and Communication. 2017.

[8] Narayanan A, Liu Yang, Chen Lihui, *et al.* Adaptive and scalable Android malware detection through online learning [C]// Proc of International Joint Conference on Neural Networks. 2016: 157-175.

[9] Feizollah A, Anuar N B, Salleh R, *et al.* AndroDialysis: analysis of Android intent effectiveness in malware detection [J]. Computers & Security, 2017, 65 (C): 121-134.

[10] Idrees F, Rajarajan M, Conti M, *et al.* PIndroid: a novel android malware detection system using ensemble learning methods [J]. Computers & Security, 2017, 68: 36-46.

[11] Xu Ke, Li Yingjiu, Deng R H. ICCDetector: ICC-based malware detection on Android [J]. IEEE Trans on Information Forensics & Security, 2016, 11 (6): 1252-1264.

[12] Yan Jinpei, Qi Yong, Rao Qifan. LSTM-based hierarchical denoising network for Android malware detection [J]. Security & Communication Networks, 2018, 2018: 1-18.

[13] Saracino A, Sgandurra D, Dini G, *et al.* MADAM: effective and efficient behavior-based Android malware detection and prevention [J]. IEEE Trans on Dependable & Secure Computing, 2018, PP (99): 1-1.

[14] Tam K, Khan S J, Fattori A, *et al.* CopperDroid: automatic reconstruction of

Android malware behaviors [C]// Proc of Network and Distributed System Security Symposium. 2015.

[15] Chen Li, Zhang Mingwei, Yang C Y, *et al.* POSTER: semi-supervised classification for dynamic Android malware detection [C]// Proc of ACM Sigsac Conference on Computer and Communications Security. 2017: 2479-2481.

[16] Zulkifli A, Hamid I R A, Shah W M, *et al.* Android malware detection based on network traffic using decision tree algorithm [C]// Proc of International Conference on Soft Computing and Data Mining. Cham: Springer, 2018: 485-494.

[17] Salehi M, Amini M. Android malware detection using Markov chain model of application behaviors in requesting system services [EB//OL]. (2017) . arXiv preprint arXiv: 1711. 05731.

[18] Narayanan A, Chandramohan M, Chen L, *et al.* A multi-view context-aware approach to Android malware detection and malicious code localization [J]. Empirical Software Engineering, 2017 (6): 1-53.

[19] Arp D, Spreitzenbarth M, Hübner M, *et al.* DREBIN: effective and explainable detection of Android malware in your pocket [C]// Proc of Network and Distributed System Security Symposium. 2014.

[20] Connor T, Ryszard W. A tool for reverse engineering Android apk files [EB/OL]. (2017-09-21) . https://ibotpeaches. github. io/Apktool/.

[21] Ed G, Christian S, Mike W, *et al.* Info-ZIP [EB/OL]. (2008-07-07) . http://www. info-zip. org/.

[22] Google company. dalvik executable format [EB/OL]. (2017-06-20) . https://source. android. com/devices/tech/dalvik/dex-format.